

Arrays Recap

Alright, I want to recap what we've learned so far about arrays, and discuss some of the common errors, that occur using arrays.

Arrays Recap

- An array is a data structure that allows us to store multiple values, of the same type, in a single variable.
- The default values of numeric array elements are set to zero.
- Arrays are zero indexed, so an array with n elements, is indexed from 0 to n - 1, for example, 10 elements would have the index range from 0 through 9.
- If we try to access an index that is out of range, Java will give us an **ArrayIndexOutOfBoundsException**, which indicates that the index is out of range, in other words, out of bounds.
- To access array elements, we use square braces. This is also known as the array access operator.

Arrays Recap - Creating a New Array

```
int[] array = new int[5];
```

This array contains the elements from array[0], through to array[4].

It has 5 elements, and an index range, from 0 to 4.

The new operator or keyword, is used to create the array, and initialize the array elements to their default values.

In this example, all the array elements will default to zero, because it is an int array.

For boolean arrays, elements would be initialized to false.

If you use String or other objects, they would be set to null, or a null reference.

Arrays Recap - Creating a new Array

We can also initialize an array inline, using an array initializer block, as we show here.

```
int[] myNumbers = {5, 4, 3, 2, 1};
```

We define the values for the array, in curly braces, in the order we want them assigned, each value separated by a comma.

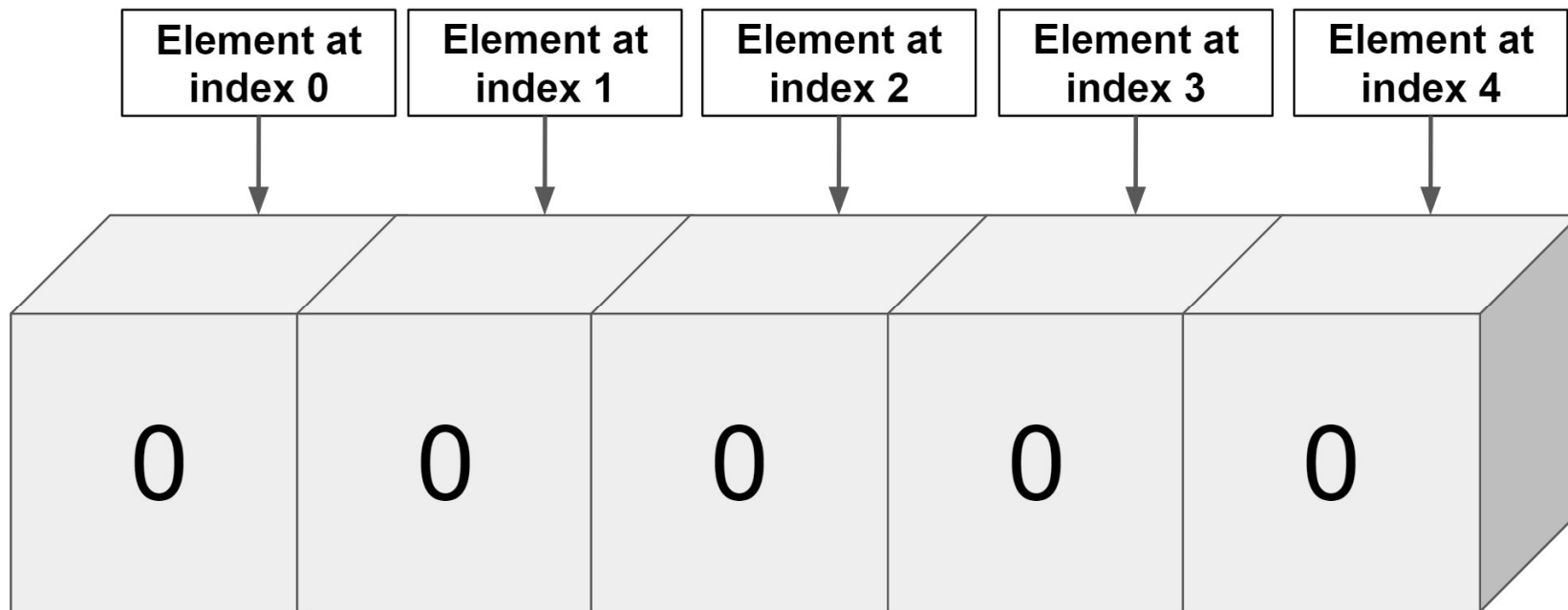
This way of initializing array elements, is also known as an anonymous array.

It has **5 elements**, and an index range from **0** to **4**.

In this example, the array elements are initialized to 5 down to 1.

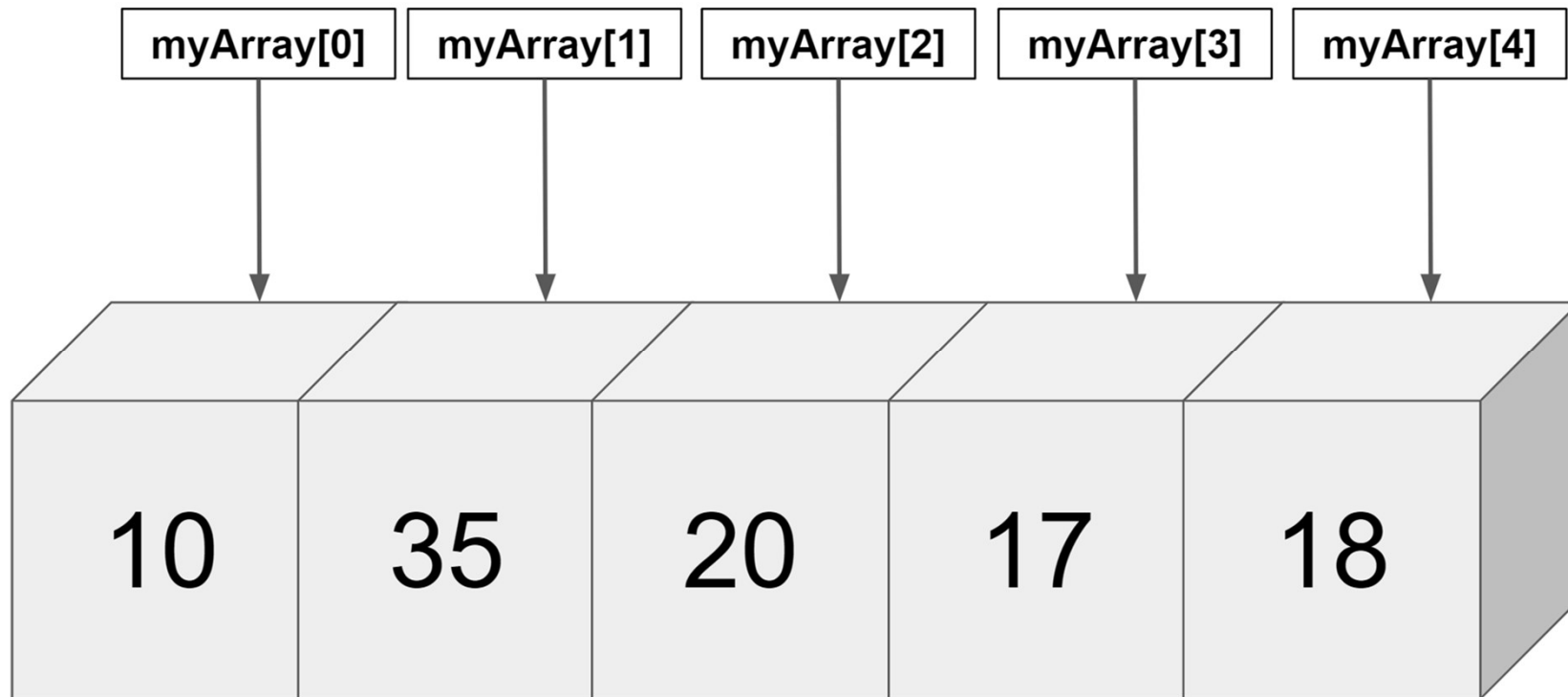
Arrays Recap

```
int[] myArray = new int[5];
```



Arrays Recap

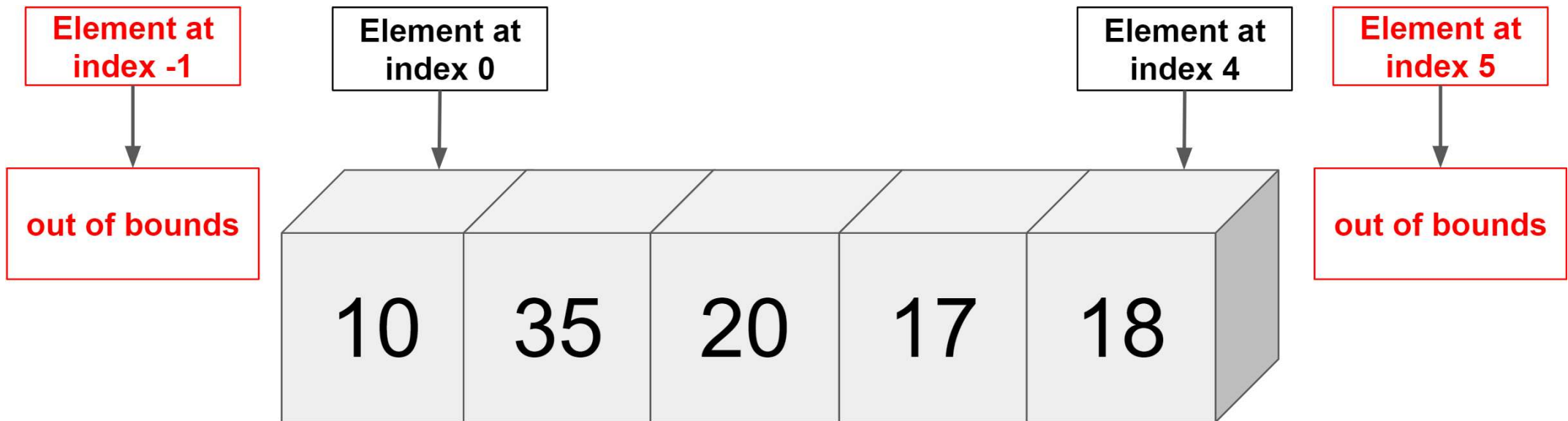
```
int[] myArray = {10, 35, 20, 17, 18};
```



First Common Error

```
int[] myArray = {10, 35, 20, 17, 18};  
myArray[5] = 55;    // out of bounds
```

- Accessing index out of range will cause error in other words **ArrayIndexOutOfBoundsException**
- We have **5 elements** and **index range is 0 to 4**



Second Common Error

```
int[] myArray = {10, 35, 20, 17, 18};  
  
for (int i = 1; i < myArray.length; i++) {  
    System.out.println("value = " + myArray[i]);  
}
```

OUTPUT:

```
value= 35  
value= 20  
value= 17  
value= 18
```


Third Common Error

```
int[] myArray = {10, 35, 20, 17, 18};

for (int i = 0; i <= myArray.length; i++) {
    System.out.println("value = " + myArray[i]);
}
```

OUTPUT:

value = 10	// printed when i = 0	-> condition 0 <= 5 is true
value = 35	// printed when i = 1	-> condition 1 <= 5 is true
value = 20	// printed when i = 2	-> condition 2 <= 5 is true
value = 17	// printed when i = 3	-> condition 3 <= 5 is true
value = 18	// printed when i = 4	-> condition 4 <= 5 is true

ArrayIndexOutOfBoundsException when i = 5 since condition 5 <= 5 is true

Third Common Error

```
int[] myArray = {10, 35, 20, 17, 18};

for (int i = 0; i < myArray.length; i++) {
    System.out.println("value = " + myArray[i]);
}
```

OUTPUT:

value = 10	// printed when i = 0	-> condition 0 <= 5 is true
value = 35	// printed when i = 1	-> condition 1 <= 5 is true
value = 20	// printed when i = 2	-> condition 2 <= 5 is true
value = 17	// printed when i = 3	-> condition 3 <= 5 is true
value = 18	// printed when i = 4	-> condition 4 <= 5 is true

Use Enhanced For Loop to avoid some of these errors

Use the enhanced for loop, if you're looping through elements from first to last, and want to process them one at a time, and you're not setting or assigning values to elements.

Enhanced For Loop (Preferred for this kind of processing)

```
int[] myArray = {10, 35, 20, 17, 18};

for (int myInt : myArray) {
    System.out.println("value = " + myInt);
}
```

Traditional For Loop

```
int[] myArray = {10, 35, 20, 17, 18};

for (int i = 0; i < myArray.length; i++) {
    System.out.println("value = " + myArray[i]);
}
```